Implementing a continuous-in-time financial API in C# *

Tarik Chakkour

▶ **To cite this version:**

Tarik Chakkour. Implementing a continuous-in-time financial API in C# *. 2016. hal-01358569

HAL Id: hal-01358569
https://hal.science/hal-01358569

Preprint submitted on 1 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Implementing a continuous-in-time financial API in *C#* *

Tarik Chakkour

Univ. Bretagne-Sud, UMR 6205, LMBA, F-56000 Vannes, France

Monday 22nd August, 2016

Abstract. In this paper, we present API for continuous-in-time financial model. This model is based on using measures and fields; and, on using mathematical operators as convolution operator. The originality of this API lie in the fact that it will be used by the company MGDIS. We describe how we impelement and check this API.

Keywords: API; mathematical computation; algorithms; discrete mathematics; software tool.

Contents

1 Introduction **2**

2 Concept of computation in API **2**

3 Implementation of measures **6**
3.1 Simple measures 6
3.2 Composed measures 7
3.2.1 Sum measure 7
3.2.2 Product measure 7
3.2.3 Piecewise measure 8
3.2.4 Truncated measure 11
3.2.5 Tabulated measure 13
3.2.6 Discrete convolution 14

4 Implementation of fields **15**
4.1 Simple fields 15
4.2 Composed fields 16
4.2.1 Sum field 16
4.2.2 Product field 16
4.2.3 Piecewise field 16
4.2.4 Truncated field 17

5 Operations between fields and measures **19**
5.1 Product of measure by a field 19
5.2 Primitive of measure defined as a field 20

6 Architecture **21**
6.1 Interfaces 21
6.2 Abstract classes 23
6.3 Unit tests 23
6.4 Math.NET Numerics 24

*This work is jointly funded by MGDIS company (`http://www.mgdis.fr/`) and the LMBA (`http://www.lmba-math.fr/`).

A Simple measures **26**

B Simple fields **26**

1 Introduction

SOFI [2] is a software tool marketed by the company MGDIS. It is designed to the public institutions such local communities to set out multiyear budgets. SOFI is based on a discrete financial modeling. Currently, the mathematical objects involved in SOFI are suites and series. The discrete model generates outcomes in the form of tables.

We showed in previous work [10] the default of this discrete model. Consequently, we build a new model with using an other paradigm in [10]. This new model is based on continuous-in-time model and uses the mathematical tools such convolution and integration. This model is based on using measures over time interval to describe loan scheme, reimbursement scheme and interest payment scheme. We checked its consistency using examples in Matlab. We refer to Frénod & Safa [11, 13, 12], which improve one of the continuous-in-time financial models built in paper [10] incorporating in it elements of control theory in order to determine the optimal loan scheme that achieves desired goals and that satisfies imposed constraints. In paper [6], we use a mathematical framework to discuss an inverse problem of determining the Loan Measure from Algebraic Spending Measure when it is possible in this model. In addition, we build a numerical method to concentrate a measure as a sum of Dirac masses.

Many problems in finance (and elsewhere) involve computing an integral. For instance, in paper [15] the Cuba library is used for multidimensional numerical integration. We describe in [18] the SBML ODE Solver Library (SOSlib) which is a programming library for symbolic and numerical analysis of chemical reaction network models. Chung, Jintai and Lee, Jang Moo propose in [7] a new family of explicit single-step time integration methods for linear and non-linear structural dynamic analyses.

In this paper we describe and software implement the continuous-in-time financial model in order to build our application programming interface (API) which is a set of routines expressed as a set of classes. The main objective of this API is to be integrated in SOFI in order to produce the continuous software tool. Consequently, it brings a competitive advantage to MGDIS. This API is restricted to certain measures and fields. Since Visual Studio is used by the mostly developers in MGDIS. The developed API was implemented in Microsoft Visual Studio 2012 $C\#$, .NET Framework 4.5.

This work proposes the implemented algorithm for our API which is shared in two librairies Lemf (Library Embedded Finance) and LemfAN (Library Embedded Finance And Numerical Analysis). In spite of LemfAN is an open-source library, Lemf and them integration in SOFI are confidential. These librairies are constitued of interfaces, classes and of abstract classes which are object classes. Since continuous-in-time financial models use financial variables which are measures and fields, the purpose of these librairies is to compute integration of measures and evaluation of fields.

The organization of the rest of this paper is as follows. Section 2 introduces time steps that are involved in the models in order to show concept of computation in API. Section 3 describes the implementation of measures. Section 4 shows implementation details about fields. Section 5 illustrates some operations between fields and measures. Section 6 focuses on some classes that presents archicture of API.

2 Concept of computation in API

This section is devoted to explain time steps that are involved in the models and the relations between them. In [14] we give the time scales to integrate density over interval. We introduce $T_{\min}$ which is the time scale below which nothing coming from the model will be observed. To be more precise, we say that a measure $\tilde{m}$ is observed over time interval $[t_1, t_2]$ if

$$\int_{t_1}^{t_2} \tilde{m}, \tag{1}$$

is computed. And, we will always, choose times t_1 and t_2 such that $t_2 - t_1 > T_{\min}$. In order to observe models, we need an observation step T_{obs} which is strictly superior than $T_{\min}$

$$T_{\text{obs}} > T_{\text{min}}. \tag{2}$$

We define the discretization step T_{dM} as a smaller step than minimal observation step T_{min} to discretize measures.

$$T_{\text{dM}} \leq T_{\text{min}}. \tag{3}$$

For instance, we are setting discrete step T_{dM} by following relation:

$$T_{\text{dM}} = \frac{T_{\text{min}}}{20}. \tag{4}$$

We define n_D as the subdivision number of observation step T_{obs} by discrete step T_{dM}

$$n_D = \left\lfloor \frac{T_{\text{obs}}}{T_{\text{dM}}} \right\rfloor. \tag{5}$$

A field is evaluated between inferior value a and superior value b with discrete step T_{dF} satisfying:

$$0 < T_{\text{dF}} < b - a. \tag{6}$$

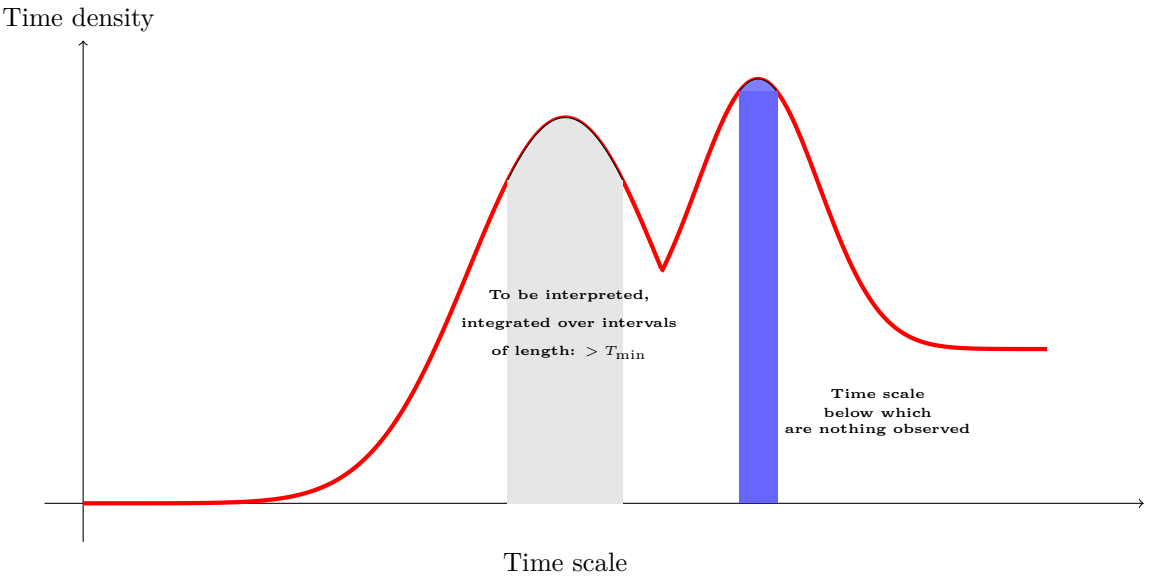

Figure 1: Different time steps.

Since measures and fields compose API, they are shared in two levels which are shown in Figure 2. The first level is called high level and is constructed for business reasons. Indeed, borrowed or repayment amounts which are computed from financial variables in high level are available for the SOFI users. The

second level is called low level which is used only by the high level. We notice that low level doesn't use its high. We say that high level implements its low.

High and low level contain non-discrete measures, non-discrete fields defined on $\mathbb{R}$, discrete measures and discrete fields. Some computations in high level need discretization. For instance, if we want to discretize a measure in high level, we construct its copy in low level and then we discretize it in order to rise up its values to high level.

Now we explain how measures are integrated and how fields are evaluated. A non-discrete measure in low level is integrated between inferior bound a and superior bound b with minimal observation step $T_{\min}$ and observation step T_{obs}. Then, discrete step T_{dM} is computed with relation (4). A non-discrete measure in high level is integrated between inferior bound a and superior bound b. Besides, a non-discrete field in low level is evaluated between inferior value a and superior value b with discrete step T_{dF}. However, the evaluation of field in high level is done between inferior value a and superior value b.

We notice that a parallelism of discretization measures and fields in low level is based on the concept of a task. Tasks provide much benefits: more efficient computation and robustness API. Indeed, the Task Parallel Library [3] is used to entail execution and development speed. We show in [17] that this library makes it easy to take advantage of potential parallelism in a program. The library relies heavily on generics and delegate expressions.

In what follows, we build the unidimensional mesh called DAS (DiscretizedAxeSegment) presented in Figure 3 for two reasons. The first reason is to better structure the low level. Since convolution operator is used in our financial model, the second reason is to compute discrete convolution. Indeed, we could not compute it with variable discrete step using the Fast Fourier Transform. Mesh DAS associated to discrete step T_{dM} is defined by a set of points $(x_k)_{k\in\mathbb{Z}}$ that are its multiple

$$\text{DAS}_{\text{T}_{\text{dM}}} = \{x_k = k \times T_{\text{dM}}, k \in \mathbb{Z}\}. \tag{7}$$

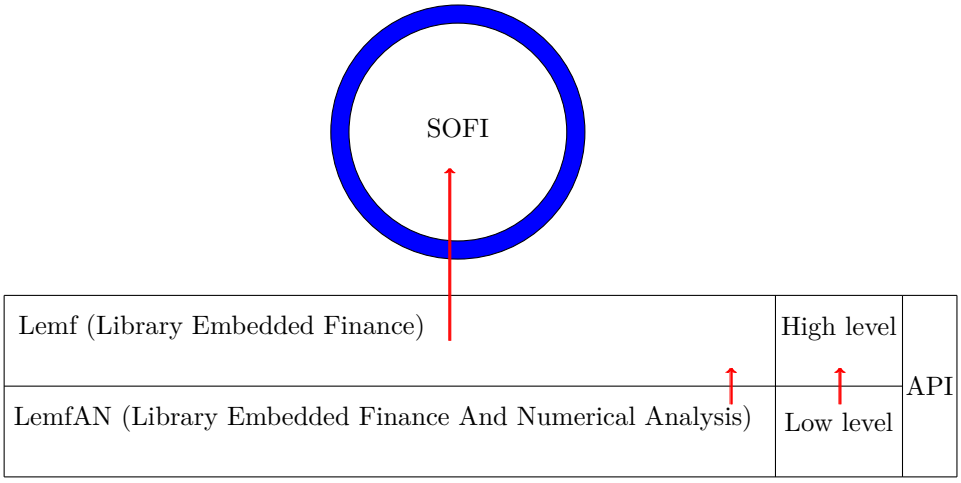

Figure 2: API composition.

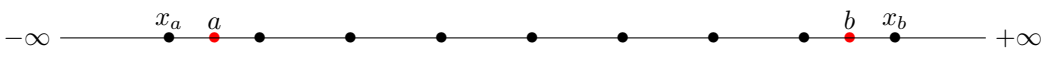

Figure 3: Mesh DAS defined on $\mathbb{R}$.

Integration of measure m_d in low level between inferior bound a and superior bound b with minimal observation step $T_{\min}$ returns its integration between new inferior bound x_a and new superior bound x_b with discrete step T_{dM}, where

$$x_a = n_a \times T_{\mathrm{dM}}, \tag{8}$$

such that:

$$n_a = \left\lfloor \frac{a}{T_{\mathrm{dM}}} \right\rfloor, \tag{9}$$

and where

$$x_b = n_b \times T_{\mathrm{dM}}, \tag{10}$$

such that:

$$n_b = \begin{cases} \frac{b}{T_{\mathrm{dM}}} & \text{if } T_{\mathrm{dM}} \text{ is divisible by } b, \\ \left\lfloor \frac{b}{T_{\mathrm{dM}}} \right\rfloor + 1 & \text{else.} \end{cases} \tag{11}$$

We define $\mathcal{N}_a^b$ by the number of subintervals of interval $[x_a, x_b]$ given by:

$$\mathcal{N}_a^b = n_b - n_a, \tag{12}$$

where, integers n_a and n_b are defined respectively in relations (9) and (11).

Now let us define a discrete measure. For any integer j from 1 to $\mathcal{N}_a^b$, we call $(n_a + j - 1)^{\mathrm{nd}}$ discrete value of measure m_d, its integration between inferior bound $(n_a + j - 1) \times T_{\mathrm{dM}}$ and superior bound $(n_a + j) \times T_{\mathrm{dM}}$

$$\forall j \in [\![1; \mathcal{N}_a^b]\!], m_d(n_a + j - 1) = \int_{(n_a+j-1)\times T_{\mathrm{dM}}}^{(n_a+j)\times T_{\mathrm{dM}}} m_d. \tag{13}$$

For any integer i from 1 to $\left\lfloor \frac{\mathcal{N}_a^b}{n_D} \right\rfloor$, we define quantity $m_d^{\mathrm{obs}}(i)$ as observed discrete measure over time interval that its length is T_{obs} between inferior bound $n_a \times T_{\mathrm{dM}} + (i-1) \times T_{\mathrm{obs}}$ and superior bound $n_a \times T_{\mathrm{dM}} + i \times T_{\mathrm{obs}}$. Formally, $m_d^{\mathrm{obs}}(i)$ is the integration of measure m_d between bounds $n_a \times T_{\mathrm{dM}} + (i-1) \times T_{\mathrm{obs}}$ and $n_a \times T_{\mathrm{dM}} + i \times T_{\mathrm{obs}}$

$$m_d^{\mathrm{obs}}(i) = \int_{n_a\times T_{\mathrm{dM}}+(i-1)\times T_{\mathrm{obs}}}^{n_a\times T_{\mathrm{dM}}+i\times T_{\mathrm{obs}}} m_d. \tag{14}$$

The integral defined in relation (14) can be decomposed with Chasles relation

$$m_d^{\mathrm{obs}}(i) = \sum_{k=1}^{k=n_D} \int_{(n_a+k-1)\times T_{\mathrm{dM}}+(i-1)\times T_{\mathrm{obs}}}^{(n_a+k-n_D)\times T_{\mathrm{dM}}+i\times T_{\mathrm{obs}}} m_d. \tag{15}$$

Because of (5) and of the fact that $l = k + (i-1) \times n_D$, the integral defined in relation (15) is approached by following integral:

$$m_d^{\text{obs}}(i) \simeq \sum_{l=1+(i-1)\times n_D}^{l=i\times n_D} \int_{(n_a+l-1)\times T_{\text{dM}}}^{(n_a+l)\times T_{\text{dM}}} m_d. \tag{16}$$

From this and according to (13), observed value $m_d^{\text{obs}}(i)$ is a sum of all values $m_d(n_a + l - 1)$ for an integer l from $1 + (i-1) \times n_D$ to $i \times n_D$

$$m_d^{\text{obs}}(i) \simeq \sum_{l=1+(i-1)\times n_D}^{l=i\times n_D} m_d(n_a + l - 1). \tag{17}$$

There are two situations of computing these observed values. The first situation consists in computing $(m_d^{\text{obs}}(i))_{1\leq i\leq \lfloor \frac{\mathcal{N}_a^b}{n_D} \rfloor}$ when $\mathcal{N}_a^b$ is divisible by n_D. The second situation consists in computing $(m_d^{\text{obs}}(i))_{1\leq i\leq \lfloor \frac{\mathcal{N}_a^b}{n_D} \rfloor+1}$ when $\mathcal{N}_a^b$ is not divisible by n_D. In these two situations, $\left\lfloor \frac{\mathcal{N}_a^b}{n_D} \right\rfloor$ observed values of measure m_d are computed with relation (17). However, in the second situation, observed value $m_d^{obs}\left(\left\lfloor \frac{\mathcal{N}_a^b}{n_D} \right\rfloor + 1 \right)$ is computed with following relation:

$$m_d^{\text{obs}}\left(\left\lfloor \frac{\mathcal{N}_a^b}{n_D} \right\rfloor + 1 \right) \simeq \sum_{k=n_D\times \lfloor \frac{\mathcal{N}_a^b}{n_D} \rfloor+1}^{k=\mathcal{N}_a^b} m_d(n_a + k - 1). \tag{18}$$

3 Implementation of measures

In this section we focus on the implementation of simple and of composed measures. We refer documents [5, 21] for integration measures. Moreover, we give the explicit integration of some simple measures. Since composed measures constitue of simple measures, we give them integrations in function of these simple measures integration.

By giving formally the integration of some measures between two bounds a and b. We easily pass to them discretizations. Indeed, the discretization of measure between inferior bound a and superior bound b with discrete step T_{dM} is its integration between points x_a and x_b which are defined in relations (8) and (10), respectively.

3.1 Simple measures

The purpose of this subsection is to define some simple measures. The simple measures can be absolutely continuous with respect to Lebesgue measure $\lambda_{\text{Lebesgue}}$ as the constant, affine, quadratic, polynomial, exponential measures or measures that are not absolutely continuous with respect to the Lebesgue measure $\lambda_{\text{Lebesgue}}$ as Dirac measures.

For instance, we created the constant measure to borrow uniformly over a time interval. For that, we define the constant density m_{Constant}, the function that is equal to a real C independently of variable time t

$$\forall t \in \mathbb{R}, m_{\text{Constant}}(t) = \text{C}. \tag{19}$$

Since constant measure $\tilde{m}_{\text{Constant}}$ is absolutely continuous with respect to Lebesgue measure $\lambda_{\text{Lebesgue}}$, it can be written in the following form:

$$\tilde{m}_{\text{Constant}} = m_{\text{Constant}}(t) \times \lambda_{\text{Lebesgue}}. \tag{20}$$

The integration of constant measure $\tilde{m}_{\text{Constant}}$ defined in relation (20) between inferior bound a and superior bound b returns $\text{C} \times (b-a)$. The table in Appendix A shows the integration of some simple measures.

3.2 Composed measures

3.2.1 Sum measure

We created sum measure in order to compute the borrowed amount of a sum of two loan measures over a time interval.

The integration of sum m of two measures m_1 et m_2 between inferior bound a and superior bound b returns a sum of two values. The first value is the integration of measure m_1 between inferior bound a and superior bound b, the second value is the integration of measure m_2 between inferior bound a and superior bound b.

The sum of two discrete measures $(m_1(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ and $(m_2(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ defined on the same universel mesh $\text{DAS}_{\text{T}_{\text{dM}}}$ is a discrete measures $m(n_a+j-1)$ of $\mathcal{N}_a^b$ values given by following relation:

$$\forall j \in [\![1;\mathcal{N}_a^b]\!], m(n_a+j-1) = m_1(n_a+j-1) + m_2(n_a+j-1). \tag{21}$$

The discretization of sum m of two non-discrete measures m_1 and m_2 in low level between inferior bound a and superior bound b is described by following algorithm:

Algorithm 1: Computation discrete measure sum

input : Measures m_1 and m_2, inferior bound a and superior bound b
output: Discrete measure $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$

- Discretize measure m_1 between points x_a and x_b of universel mesh $\text{DAS}_{\text{T}_{\text{dM}}}$ to get discrete measure $(m_1(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$;
- Discretize measure m_2 between points x_a and x_b of universel mesh $\text{DAS}_{\text{T}_{\text{dM}}}$ to get discrete measure $(m_2(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$;
- Use relation (21) to get discrete measure $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ which is the discretization of measure m between points x_a and x_b ;

return $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$;

3.2.2 Product measure

We define the product of two measures for reasons of nature software production. This product is not used in continuous-in-time financial model. In addition, since the product of two Dirac measures has no sense in measure theory, we prohibit this type of product.

The product of two discrete measures $(m_1(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ and $(m_2(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ defined on the same universel mesh is discrete measure $m(n_a+j-1)$ of $\mathcal{N}_a^b$ values given by following relation:

$$\forall j \in [\![1;\mathcal{N}_a^b]\!], m(n_a+j-1) = \frac{m_1(n_a+j-1) \times m_2(n_a+j-1)}{T_{\text{dM}}}. \tag{22}$$

The integration of discrete measure $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ defined in relation (22) between inferior bound a and superior bound b is the sum of all values $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$

$$\sum_{j=1}^{j=\mathcal{N}_a^b} m(n_a+j-1). \tag{23}$$

The consistency of relation (22) is illustrated using following example. This exemple consists in computing the discrete product of two discrete measures defined on the same universel mesh $\mathrm{DAS}_{\mathrm{T_{dM}}}$. The first discrete measure $(m_1(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ is defined by discretization of constant measure that constant equals to C_1, between inferior bound x_a and superior bound x_b with discrete step T_{dM}

$$\forall j \in [\![1;\mathcal{N}_a^b]\!], m_1(n_a+j-1) = \mathrm{C}_1 \times T_{\mathrm{dM}}, \tag{24}$$

and the second discrete measure $(m_2(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ is defined by discretization of constant measure that constant equals to C_2, between inferior bound x_a and superior bound x_b with discrete step T_{dM}

$$\forall j \in [\![1;\mathcal{N}_a^b]\!], m_2(n_a+j-1) = \mathrm{C}_2 \times T_{\mathrm{dM}}. \tag{25}$$

Next, relation (22) is used for computing discrete product $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ to obtain:

$$\forall j \in [\![1;\mathcal{N}_a^b]\!], m(n_a+j-1) = \mathrm{C}_1 \times \mathrm{C}_2 \times T_{\mathrm{dM}}. \tag{26}$$

Now, if we descretize constant measure that constant equals to $\mathrm{C}_1 \times \mathrm{C}_2$ between two points x_a and x_b of universel mesh $\mathrm{DAS}_{\mathrm{T_{dM}}}$, then we get the same discrete values defined in relation (26). It is concluded that relation (22) is consistent.

Since we expressed previously the product of two discrete measures and its integration, we want to integrate a product measure in high level. Indeed, the integration of product m of two measures m_1 and m_2 between inferior bound a and superior bound b with minimal observation step $T_{\min}$ is given by following algorithm:

Algorithm 2: Integration algorithm of product m of two measures m_1 and m_2 in high level

input : Measures m_1 and m_2, inferior bound a and superior bound b
output: Integration value v

- Compute discrete step T_{dM} from minimal observation step $T_{\min}$ with relation (4) ;
- Descretize measure m_1 between two points x_a and x_b of universel mesh $\mathrm{DAS}_{\mathrm{T_{dM}}}$ to get discrete measure $(m_1(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$;
- Descretize measure m_2 between two points x_a and x_b of universel mesh $\mathrm{DAS}_{\mathrm{T_{dM}}}$ to get discrete measure $(m_2(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$;
- Compute with relation (22) discrete product measure $(m_(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ of two discrete measures $(m_1(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ and $(m_2(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$;
- Integrate discrete measure $(m(n_a+j-1))_{1\leq j\leq \mathcal{N}_a^b}$ using relation (23) to obtain v ;

return v ;

3.2.3 Piecewise measure

We call $\tilde{m}_{\mathrm{Piecewise}}$ a piecewise measure, one of the two construction methods defined

• from a real Fr_0 allowing to generate measures m_0 and m_1 respectively on intervals $]-\infty, Fr_0]$ and $[Fr_0, +\infty[$. Formally, measure $\tilde{m}_{\mathrm{Piecewise}}$ presented in Figure 4 is a piecewise measure on $\mathbb{R}$ if and only if

$$\exists Fr_0 \in \mathbb{R}, \text{ such that: } \tilde{m}_{\mathrm{Piecewise}}\,|_{]-\infty,Fr_0]} = m_0, \tilde{m}_{\mathrm{Piecewise}}\,|_{[Fr_0,+\infty[} = m_1, \\ \text{where } m_0 \text{ and } m_1 \text{ are any measures.} \tag{27}$$

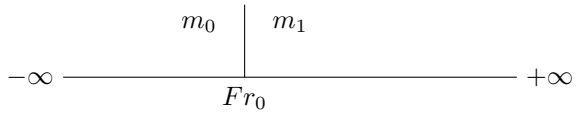

Figure 4: Piecewise measure $\tilde{m}_{\text{Piecewise}}$ constitued of measures m_0, m_1 and frontier Fr_0.

• from a subdivision $(Fr_0, Fr_1, \ldots, Fr_n)$ of $n+2$ intervals allowing to generate the measure m_i on each closed interval $[Fr_{i-1}, Fr_i]$ for i from 1 to n and to generate both measures m_0 and m_{n+1} respectively on the two intervals $]-\infty, Fr_0]$ and $[Fr_n, +\infty[$. Formally, measure $\tilde{m}_{\text{Piecewise}}$ presented in Figure 5 is a piecewise measure on $\mathbb{R}$ if and only if

$$\begin{gathered}\exists (Fr_0, Fr_1, \ldots, Fr_n), \;\; Fr_0 < Fr_1 < \cdots < Fr_n \;\text{ such that: }\; \forall i \in \{1, 2, \ldots, n\} \\ \tilde{m}_{\text{Piecewise}}\,|_{]-\infty, Fr_0]} = m_0, \tilde{m}_{\text{Piecewise}}\,|_{[Fr_{i-1}, Fr_i]} = m_i, \tilde{m}_{\text{Piecewise}}\,|_{[Fr_n, +\infty[} = m_{n+1}, \\ \text{where } m_0, \; m_i \text{ and } m_{n+1} \text{ are any measures.}\end{gathered} \tag{28}$$

Figure 5: Piecewise measure $\tilde{m}_{\text{Piecewise}}$ constitued of a list of measures $(m_i)_{0 \leq i \leq n+1}$ and of a list of frontiers $(Fr_i)_{0 \leq i \leq n}$.

The integration of piecewise measure $\tilde{m}_{\text{Piecewise}}$ defined in relation (27) is given by following algorithm:

Algorithm 3: Integration algorithm of piecewise measure $\tilde{m}_{\text{Piecewise}}$ defined in relation (27)

input : Measures m_0 and m_1, frontier Fr_0, inferior bound a and superior bound b
output: Integration value v
if $b \leq Fr_0$ **then**
  v is the integration of measure m_0 between inferior bound a and superior bound b ;
else if $Fr_0 \leq a$ **then**
  v is the integration of measure m_1 between inferior bound a and superior bound b ;
else
  v is the sum of two quantities, where the first quantity is the integration of measure m_0 between inferior bound a and superior bound Fr_0, and the second quantity is the integration of measure m_1 between inferior bound Fr_0 and superior bound b ;
return v ;

In order to integrate the piecewise measure $\tilde{m}_{\text{Piecewise}}$ defined in relation (28) between inferior bound a and superior bound b, we define index p and q respectively by the index for first and last measures of $(m_i)_{0 \leq i \leq n+1}$ to be integrated. These index p et q are determined by dichotomy search. By considering $l \in [\![1; n]\!]$, index $p, q \in [\![0; n+1]\!]$ are defined using a list of frontiers $(Fr_i)_{0 \leq i \leq n}$ as:

$$p = \begin{cases} 0 & \text{if } a < Fr_0, \\ l & \text{if } Fr_{l-1} \leq a < Fr_l, \\ n+1 & \text{if } Fr_n \leq a. \end{cases} \tag{29}$$

$$q = \begin{cases} 0 & \text{if } b \leq Fr_0, \\ l & \text{if } Fr_{l-1} < b \leq Fr_l, \\ n+1 & \text{if } Fr_n < b. \end{cases} \tag{30}$$

It is necessary to define variables $a_\star$ and $b_\star$ in order to integrate generally the piecewise measure $\tilde{m}_{\text{Piecewise}}$. Variable $a_\star$ means the superior integration bound of measure m_p. If $a < Fr_n$, then $a_\star$ is equal to the inferior value of Fr_p and of b, if $Fr_n \leq a$, then $a_\star$ is equal to b. Formally, variable $a_\star$ is defined as:

$$a_\star = \begin{cases} \inf\{Fr_p, b\} & \text{if } a < Fr_n, \\ b & \text{if } Fr_n \leq a. \end{cases} \tag{31}$$

Variable $b_\star$ means the inferior integration bound of measure m_q. Similary, if $Fr_0 < b$, then $b_\star$ is equal to the superior value of Fr_{q-1} and of a, if $b \leq Fr_0$, then $b_\star$ is equal to a. Formally, variable $b_\star$ is defined as:

$$b_\star = \begin{cases} \sup\{Fr_{q-1}, a\} & \text{if } Fr_0 < b, \\ a & \text{if } b \leq Fr_0. \end{cases} \tag{32}$$

Moreover, we define quantities q_1, q_2 and q_3 as follows. The quantity q_1 is the integration of measure m_p between inferior bound a and superior bound $a_\star$:

$$q_1 = \int_a^{a_\star} m_p. \tag{33}$$

The quantity q_2 is the sum of integration of measure m_{i+1} between inferior bound Fr_i and superior bound Fr_{i+1} for i from p to $q-2$:

$$q_2 = \sum_{i=p}^{i=q-2} \int_{Fr_i}^{Fr_{i+1}} m_{i+1}. \tag{34}$$

The quantity q_3 is the integration of measure m_q between inferior bound $b_\star$ and superior bound b:

$$q_3 = \int_{b_\star}^{b} m_q. \tag{35}$$

Consequently, the algorithm of integrating piecewise measure $\tilde{m}_{\text{Piecewise}}$ defined in relation (28) is given as follows:

Algorithm 4: Integration algorithm of piecewise measure $\tilde{m}_{\text{Piecewise}}$ defined in relation (28)

input : List of measures $(m_i)_{0 \leq i \leq n+1}$, list of frontiers $(Fr_i)_{0 \leq i \leq n}$, inferior bound a and superior bound b
output: Integration value v
if $p = q$ **then**
  $v \leftarrow q_1$;
else if $p = q - 1$ **then**
  $v \leftarrow q_1 + q_3$;
else
  $v \leftarrow q_1 + q_2 + q_3$;
return v;

3.2.4 Truncated measure

We call $\tilde{m}_{\text{Truncated}}$ a truncated measure, one of the three construction methods defined

• from a subdivision (Fr_0, Fr_1) of 3 intervals allowing to generate the measure m_1 on interval $[Fr_0, Fr_1]$ and to generate the null measures m_0 and m_2 respectively on intervals $]-\infty, Fr_0]$ and $[Fr_1, +\infty[$. Formally, measure $\tilde{m}_{\text{Truncated}}$ presented in Figure 6 is a truncated measure on $\mathbb{R}$ if and only if

$$\begin{aligned}\exists (Fr_0, Fr_1),\ \ Fr_0 < Fr_1 \ \text{ such that: } \tilde{m}_{\text{Truncated}}\,|_{]-\infty, Fr_0]} = m_0, \tilde{m}_{\text{Truncated}}\,|_{[Fr_0, Fr_1]} = m_1,\\ \tilde{m}_{\text{Truncated}}\,|_{[Fr_1, +\infty[} = m_2, \text{where } m_0,\ m_2 \text{ are null measures and } m_1 \text{ is any measure.}\end{aligned} \tag{36}$$

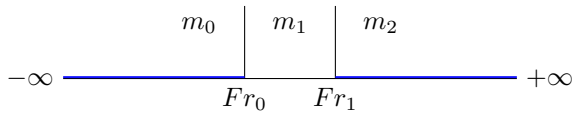

Figure 6: Truncated measure $\tilde{m}_{\text{Truncated}}$ constitued of measure m_1, null measures m_0, m_2 and of frontiers Fr_0 and Fr_1.

• from a real Fr_0 allowing to generate the null measure m_0 on interval $]-\infty, Fr_0]$ and to generate the measure m_1 on interval $[Fr_0, +\infty[$. Formally, measure $\tilde{m}_{\text{Truncated}}$ presented in Figure 7 is a truncated measure on $\mathbb{R}$ if and only if

$$\begin{aligned}\exists Fr_0 \in \mathbb{R},\ \text{ such that: } \tilde{m}_{\text{Truncated}}\,|_{]-\infty, Fr_0]} = m_0, \tilde{m}_{\text{Truncated}}\,|_{[Fr_0, +\infty[} = m_1,\\ \text{where } m_0 \text{ is null measure and } m_1 \text{ is any measure.}\end{aligned} \tag{37}$$

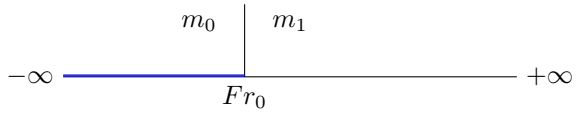

Figure 7: Truncated measure $\tilde{m}_{\text{Truncated}}$ constitued of null measure m_0, measure m_1 and of frontier Fr_0.

• from a real Fr_1 allowing to generate the measure m_1 on interval $]-\infty, Fr_1]$ and to generate the measure m_2 on interval $[Fr_1, +\infty[$. Formally, measure $\tilde{m}_{\text{Truncated}}$ presented in Figure 8 is a truncated measure on $\mathbb{R}$ if and only if

$$\begin{aligned}\exists Fr_1 \in \mathbb{R}\ \text{ such that: } \tilde{m}_{\text{Truncated}}\,|_{]-\infty, Fr_1]} = m_1, \tilde{m}_{\text{Truncated}}\,|_{[Fr_1, +\infty[} = m_2,\\ \text{where } m_1 \text{ is any measure and } m_2 \text{ is null measure.}\end{aligned} \tag{38}$$

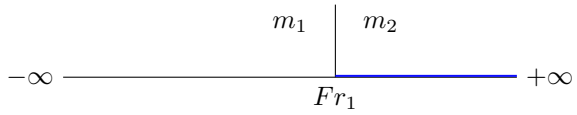

Figure 8: Truncated measure $\tilde{m}_{\text{Truncated}}$ constitued of measure m_1, null measure m_2 and of frontier Fr_1.

The algorithm of integrating truncated measure $\tilde{m}_{\text{Truncated}}$ defined in relation (36) between inferior bound a and superior bound b is given as follows:

Algorithm 5: Integration algorithm of truncated measure $\tilde{m}_{\text{Truncated}}$ defined in relation (36)

input : Measures m_0, m_1 and m_2, frontiers Fr_0 and Fr_1, inferior bound a and superior bound b
output: Integration value v
if $a < Fr_0 < Fr_1 < b$ **then**
  v is the integration of measure m_1 between inferior bound Fr_0 and superior bound Fr_1 ;
else if $Fr_0 \leq a < b \leq Fr_1$ **then**
  v is the integration of measure m_1 between inferior bound a and superior bound b ;
else if $Fr_0 \leq a < Fr_1 < b$ **then**
  v is the integration of measure m_1 between inferior bound a and superior bound Fr_1 ;
else if $a < Fr_0 < b \leq Fr_1$ **then**
  v is the integration of measure m_1 between inferior bound Fr_0 and superior bound b ;
else
  v is zero ;
return v ;

The algorithm of integrating truncated measure $\tilde{m}_{\text{Truncated}}$ defined in relation (37) between inferior bound a and superior bound b is given as follows:

Algorithm 6: Integration algorithm of truncated measure $\tilde{m}_{\text{Truncated}}$ defined in relation (37)

input : Measures m_0 and m_1, frontier Fr_0, inferior bound a and superior bound b
output: Integration value v
if $a < Fr_0 < b$ **then**
  v is the integration of measure m_1 between inferior bound Fr_0 and superior bound b ;
else if $Fr_0 \leq a$ **then**
  v is the integration of measure m_1 between inferior bound a and superior bound b ;
else
  v is zero ;
return v ;

The algorithm of integrating truncated measure $\tilde{m}_{\text{Truncated}}$ defined in relation (38) between inferior bound a and superior bound b is given as follows:

Algorithm 7: Integration algorithm of truncated measure $\tilde{m}_{\text{Truncated}}$ defined in relation (38)

input : Measures m_1 and m_2, frontier Fr_1, inferior bound a and superior bound b
output: Integration value v
if $a < Fr_1 < b$ **then**
  v is the integration of measure m_1 between inferior bound a and superior bound Fr_1 ;
else if $b \leq Fr_1$ **then**
  v is the integration of measure m_1 between inferior bound a and superior bound b ;
else
  v is zero ;
return v ;

3.2.5 Tabulated measure

The aim of this subsubsection is to define a tabulated measure and to give its integration algorithm. There are three reasons for the creation of tabulated measure. The first one is to translate an array of values to a discrete measure. The second one is to compute the convolution measure. The third one is to create a discrete measure in low level in order to transfer it to high level.

We want to build a tabulated measure $\tilde{m}_{\text{Tabulated}}$ between inferior value V_I and superior bound V_S strictly superior than V_I with a set of n values $(l_i)_{0\leq i\leq n-1}$. For that, we use a tabulation step T_{tab} in order to share values $(l_i)_{0\leq i\leq n-1}$ between V_I and V_S, defined by:

$$T_{\text{tab}} = \frac{V_S - V_I}{n}. \tag{39}$$

We call $\tilde{m}_{\text{Tabulated}}$ a tabulated measure, a construction method defined from a subdivision $(Fr_0, Fr_1, \ldots, Fr_n)$ given by following frontiers:

$$\forall i \in [\![0; n]\!], Fr_i = V_I + i \times T_{\text{tab}}, \tag{40}$$

allowing to generate the constant density m_{j+1} on each closed interval $[Fr_j, Fr_{j+1}]$ for j from 0 to $n-1$ and to generate both null densities m_0 and m_{n+1} respectively on two intervals $]-\infty, Fr_0]$ and $[Fr_n, +\infty[$. Tabulated measure $\tilde{m}_{\text{Tabulated}}$ is illustrated in Figure 9.

Figure 9: Tabulated measure $\tilde{m}_{\text{Tabulated}}$ constitued of densities $(m_i)_{0\leq i\leq n+1}$ and of frontiers $(Fr_i)_{0\leq i\leq n}$ such that m_0 and m_{n+1} are null densities and such that each density m_{j+1} is a constant density defined on closed interval $[Fr_j, Fr_{j+1}]$ that constant equals to $\frac{l_j}{T_{\text{tab}}}$ for j from 0 to $n-1$.

To integrate tabulated measure $\tilde{m}_{\text{Tabulated}}$ explicitly between inferior bound a and superior bound b, index p and q given respectively in relations (29) and (30) are used. Since, tabulated measure $\tilde{m}_{\text{Tabulated}}$ is constitued of null densities m_0 and m_{n+1}, we describe its in algorithm 8.

Algorithm 8: Integration algorithm of tabulated measure $\tilde{m}_{\text{Tabulated}}$

input : Values $(l_i)_{0\leq i\leq n-1}$, inferior value V_I, superior value V_S, inferior bound a and superior bound b

output: Integration value v

if $a < V_I < V_S < b$ **then**

$$v \leftarrow \sum_{i=0}^{i=n-1} l_i \ ;$$

else if $V_I \leq a < V_S < b$ **then**

if $Fr_{n-1} \leq a < V_S$ **then**

$$v \leftarrow (V_S - a) \times \frac{l_{n-1}}{T_{\text{tab}}} \ ;$$

else

$$v \leftarrow (Fr_p - a) \times \frac{l_{p-1}}{T_{\text{tab}}} + \sum_{i=p}^{i=n-1} l_i \ ;$$

else if $a < V_I < b \leq V_S$ **then**

if $V_I < b \leq Fr_1$ **then**

$$v \leftarrow (b - V_I) \times \frac{l_0}{T_{\text{tab}}} \ ;$$

else

$$v \leftarrow \sum_{i=0}^{i=q-2} l_i + (b - Fr_{q-1}) \times \frac{l_{q-1}}{T_{\text{tab}}} \ ;$$

else if $V_I \leq a < b \leq V_S$ **then**

if $p = q$ **then**

$$v \leftarrow (b - a) \times \frac{l_{p-1}}{T_{\text{tab}}} \ ;$$

else if $p = q - 1$ **then**

$$v \leftarrow (Fr_p - a) \times \frac{l_{p-1}}{T_{\text{tab}}} + (b - Fr_p) \times \frac{l_p}{T_{\text{tab}}} \ ;$$

else

$$v \leftarrow (Fr_p - a) \times \frac{l_{p-1}}{T_{\text{tab}}} + \sum_{i=p}^{i=q-2} l_i + (b - Fr_{q-1}) \times \frac{l_{q-1}}{T_{\text{tab}}} \ ;$$

else

$v \leftarrow 0$;

return v;

3.2.6 Discrete convolution

Discrete convolution is a fundamental operation in the financial model. Indeed, loan measure $\tilde{\kappa}_E$ and capital repayment measure $\tilde{\rho}_{\mathcal{K}}$ are connected by a convolution operator. It is necessary to implement it in order to compute repayment amount. Then the discrete convolution may be evaluated with the aid of the Fast Fourier Transform (FFT) method.

We refer to papers [16, 20], which are dealing with how convolution can be efficiently computed by FFT. For example, algorithms based on explicit computation and on the Fast Fourier Transform are described in [16]. Paper [20] presents a more efficient computation of the convolution between a compressed text and an uncompressed pattern.

By the Fourier convolution theorem, the discrete Fourier transform of $\tilde{\kappa}_E \star \tilde{\gamma}$ may be computed as

$$\mathcal{F}(\tilde{\rho}_{\mathcal{K}}) = \mathcal{F}(\tilde{\kappa}_E \star \tilde{\gamma}) = \mathcal{F}(\tilde{\kappa}_E) \bullet \mathcal{F}(\tilde{\gamma}), \tag{41}$$

where the repayment pattern measure $\tilde{\gamma}$ expresses the way an amount 1 borrowed at $t = 0$ is repaid and where $\bullet$ denotes component-wise multiplication, and $\mathcal{F}(\tilde{\kappa}_E)$ and $\mathcal{F}(\tilde{\gamma})$ discrete Fourier transforms

of $\tilde{\kappa}_E$ and $\tilde{\gamma}$, respectively. The aim here is to compute discrete convolution $(\tilde{\kappa}_E \star \tilde{\gamma}(n_e + j - 1))_{1 \leq j \leq \mathcal{N}_e^f}$ of discrete measures $(\tilde{\kappa}_E(n_a + j - 1))_{1 \leq j \leq \mathcal{N}_a^b}$ and $(\tilde{\gamma}(n_c + j - 1))_{1 \leq j \leq \mathcal{N}_c^d}$ between points x_e and x_f of universel mesh $\mathrm{DAS}_{\mathrm{T_{dM}}}$. The computations are summarized in Algorithm 9.

Algorithm 9: Computation discrete convolution with FFT

input : Measures $\tilde{\kappa}_E$ and $\tilde{\gamma}$, inferior bound e and superior bound f
output: $(\tilde{\kappa}_E \star \tilde{\gamma}(n_e + j - 1))_{1 \leq j \leq \mathcal{N}_e^f}$

- Determine the convex hull of the support of discrete measure $(\tilde{\kappa}_E(n_a + j - 1))_{1 \leq j \leq \mathcal{N}_a^b}$ supposed to be interval $[x_{a_1}, x_{b_1}]$;
- Determine the convex hull of the support of discrete measure $(\tilde{\gamma}(n_c + j - 1))_{1 \leq j \leq \mathcal{N}_c^d}$ supposed to be interval $[x_{c_1}, x_{d_1}]$;
- Complete by zero discrete measures $(\tilde{\kappa}_E(n_a + j - 1))_{1 \leq j \leq \mathcal{N}_a^b}$ and $(\tilde{\gamma}(n_c + j - 1))_{1 \leq j \leq \mathcal{N}_c^d}$ such that they have N values, where N is power of 2 and is smallest value satisfying $N \geq \mathcal{N}_a^b + \mathcal{N}_c^d$. Then, $(\tilde{\kappa}_E^1(n_a + j - 1))_{1 \leq j \leq N}$ and $(\tilde{\gamma}^1(n_c + j - 1))_{1 \leq j \leq N}$ are called the discrete values extended by zero ;
- Compute discrete measures $(x(n_a + j - 1))_{1 \leq j \leq N}$ and $(y(n_c + j - 1))_{1 \leq j \leq N}$ by Fourier transform of discrete measures $(\tilde{\kappa}_E^1(n_a + j - 1))_{1 \leq j \leq N}$ and $(\tilde{\gamma}^1(n_c + j - 1))_{1 \leq j \leq N}$, respectively ;
- Compute vector $z(j-1)_{1 \leq j \leq N}$ defined by element-wise multiplication of $(x(n_a + j - 1))_{1 \leq j \leq N}$ by $(y(n_c + j - 1))_{1 \leq j \leq N}$;
- Compute vector $(h(j-1))_{1 \leq j \leq N}$ defined by inverse Fourier transform of $(z(j-1))_{1 \leq j \leq N}$;
- Construct tabulated measure $\tilde{m}_{\mathrm{Tabulated}}$ between inferior value $x_{a_1} + x_{c_1}$ and superior value $x_{b_1} + x_{d_1}$ with a set of first $\mathcal{N}_a^b + \mathcal{N}_c^d$ values of h ;
- Discretize tabulated measure $\tilde{m}_{\mathrm{Tabulated}}$ between points x_e et x_f with discrete step T_{dM} to get discrete values $(\tilde{\kappa}_E \star \tilde{\gamma}(n_e + j - 1))_{1 \leq j \leq \mathcal{N}_e^f}$;

return $(\tilde{\kappa}_E \star \tilde{\gamma}(n_e + j - 1))_{1 \leq j \leq \mathcal{N}_e^f}$;

4 Implementation of fields

We define field as continuous function by superior value. Fields are shared in two categories which are simple and composed. In what follows, we give them definitions and hwo they are evaluated in high level at point. Consequently, the discretization of non-discrete fields in low level is based on them evaluation. Indeed, its discretization between inferior value a and superior value b with discrete step T_{dF} returns its evaluation between points x_a and x_b which are defined in relations (8) and (10), respectively.

4.1 Simple fields

The aim of this subsection is to define some simple fields which are constant, affine, quadratic, polynomial, and exponential fields.

For instance, we created the constant field in order to compute the borrowed amount at a given instant where the loan is a constant function. The constant field F_{Constant} is defined as the function that is equal to C independently of variable time t.

$$\forall t \in \mathbb{R}, F_{\mathrm{Constant}}(t) = \mathrm{C}. \tag{42}$$

The evaluation of constant field F_{Constant} defined in relation (42) returns constant C. The table in Appendix B shows the evaluation of some simple fields.

4.2 Composed fields

This subsection is entirely devoted to define some composed fields which are sum, product, piecewise, and truncated fields.

4.2.1 Sum field

We created a sum field in order to compute the sum of two fields at an instant t. For example, we can compute the sum of two current debts at this time t.

The evaluation of the sum F of two fields F_1 and F_2 in high level at instant t returns a value which is the sum of two values. The first value is the evaluation of field F_1 at time t, the second value is the evaluation of field F_2 at time t. The evaluation of field F is expressed by following relation:

$$\forall t \in \mathbb{R}, \mathrm{F}(t) = \mathrm{F}_1(t) + \mathrm{F}_2(t). \tag{43}$$

4.2.2 Product field

Since we define previously the sum field in subsubsection 4.2.1, we define by the same way the product F of two fields F_1 and F_2 in high level which is given by following relation:

$$\forall t \in \mathbb{R}, \mathrm{F}(t) = \mathrm{F}_1(t) \times \mathrm{F}_2(t). \tag{44}$$

4.2.3 Piecewise field

We call $\mathrm{F}_{\mathrm{Piecewise}}$ a piecewise field, one of the two construction methods defined

• from a real Fr_0 allowing to generate fields F_0 and F_1 respectively on intervals $]-\infty, Fr_0]$ and $[Fr_0, +\infty[$. Formally, field $\mathrm{F}_{\mathrm{Piecewise}}$ illustrated in Figure 10 is a piecewise field on $\mathbb{R}$ if and only if

$$\begin{gathered}\exists Fr_0 \in \mathbb{R}, \text{ such that: } \mathrm{F}_{\mathrm{Piecewise}}|_{]-\infty, Fr_0]} = F_0, \mathrm{F}_{\mathrm{Piecewise}}|_{[Fr_0, +\infty[} = F_1, \\ \text{where } F_0 \text{ and } F_1 \text{ are any fields.}\end{gathered} \tag{45}$$

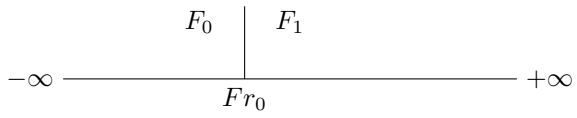

Figure 10: Piecewise field $\mathrm{F}_{\mathrm{Piecewise}}$ constitued of fields F_0, F_1 and of frontier Fr_0.

• from a subdivision $(Fr_0, Fr_1, \ldots, Fr_n)$ of $n+2$ intervals allowing to generate the field F_i on each closed interval $[Fr_{i-1}, Fr_i]$ for i from 1 to n and to generate both fields F_0 and F_{n+1} respectively on two intervals $]-\infty, Fr_0]$ and $[Fr_n, +\infty[$. Formally, field $\mathrm{F}_{\mathrm{Piecewise}}$ presented in Figure 11 is a piecewise field on $\mathbb{R}$ if and only if

$$\begin{gathered}\exists (Fr_0, Fr_1, \ldots, Fr_n), \ , Fr_0 < Fr_1 < \cdots < Fr_n \text{ such that: } \forall i \in \{1, 2, \ldots, n\} \\ \mathrm{F}_{\mathrm{Piecewise}}|_{[Fr_{i-1}, Fr_i]} = F_i, \mathrm{F}_{\mathrm{Piecewise}}|_{]-\infty, Fr_0]} = F_0, \mathrm{F}_{\mathrm{Piecewise}}|_{[Fr_n, +\infty[} = F_{n+1}, \\ \text{where } F_i, \ F_0, \ F_{n+1} \text{ are any fields.}\end{gathered} \tag{46}$$

Figure 11: Piecewise field $\mathrm{F}_{\mathrm{Piecewise}}$ constitued of a list of fields $(F_i)_{0\leq i\leq n+1}$ and of a list of frontiers $(Fr_i)_{0\leq i\leq n}$.

The evaluation of piecewise field $\mathrm{F}_{\mathrm{Piecewise}}$ defined in relation (45) at instant t is given by following algorithm:

Algorithm 10: Evaluation algorithm of piecewise field $\mathrm{F}_{\mathrm{Piecewise}}$ defined in relation (45)

input : Fields F_0 and F_1, frontier Fr_0, inferior bound a and superior bound b
output: Evaluation value v
if $t < Fr_0$ **then**
  v is the evaluation of field F_0 at instant t ;
else
  v is the evaluation of field F_1 at instant t ;
return v ;

We give an algorithm for the evaluation of piecewise field $\mathrm{F}_{\mathrm{Piecewise}}$ defined in relation (46) at instant t as follows:

Algorithm 11: Evaluation algorithm of piecewise field $\mathrm{F}_{\mathrm{Piecewise}}$ defined in relation (46)

input : List of fields $(F_i)_{0\leq i\leq n+1}$, list of frontiers $(Fr_i)_{0\leq i\leq n}$, inferior bound a and superior bound b
output: Integration value v
if $t < Fr_0$ **then**
  v is the evaluation of field F_0 at instant t ;
else if $Fr_i \leq t < Fr_{i+1}$ **then**
  v is the evaluation of field F_{i+1} at instant t ;
else if $t \geq Fr_n$ **then**
  v is the evaluation of field F_{n+1} at instant t ;
return v ;

4.2.4 Truncated field

We call $\mathrm{F}_{\mathrm{Truncated}}$ a truncated field, one of the three construction methods defined

• from a subdivision (Fr_0, Fr_1) of 3 intervals allowing to generate the field F_1 on interval $[Fr_0, Fr_1]$ and to generate both null fields F_0 and F_2 respectively on intervals $]-\infty, Fr_0]$ and $[Fr_1, +\infty[$. Formally, field $\mathrm{F}_{\mathrm{Truncated}}$ presented in Figure 12 is a truncated field on $\mathbb{R}$ if and only if

$$\begin{aligned}\exists(Fr_0, Fr_1),\ \ Fr_0 < Fr_1 \ \text{ such that: } \mathrm{F}_{\mathrm{Truncated}}|_{]-\infty,Fr_0]} = F_0, \mathrm{F}_{\mathrm{Truncated}}|_{[Fr_0,Fr_1]} = F_1,\\ \mathrm{F}_{\mathrm{Truncated}}|_{[Fr_1,+\infty[} = F_2, \text{where } F_0 \text{ and } F_2 \text{ are null fields and } F_1 \text{ is any field.}\end{aligned} \tag{47}$$

• from a real Fr_0 allowing to generate the null field F_0 on interval $]-\infty, Fr_0]$ and to generate the field F_1 on interval $[Fr_0, +\infty[$. Formally, field $\mathrm{F}_{\mathrm{Truncated}}$ presented in Figure 13 is a truncated field on $\mathbb{R}$ if and only if

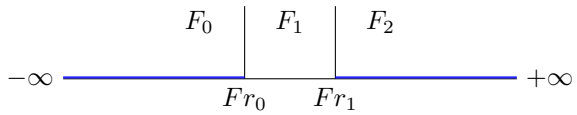

Figure 12: Truncated field $\mathrm{F}_{\mathrm{Truncated}}$ constitued of field F_1 and of null fields F_0, F_2 and of frontiers Fr_0 and Fr_1.

$$\exists Fr_0 \in \mathbb{R}, \text{ such that: } \mathrm{F}_{\mathrm{Truncated}} |_{]-\infty, Fr_0]} = F_0, \mathrm{F}_{\mathrm{Truncated}} |_{[Fr_0, +\infty[} = F_1, \\ \text{where } F_0 \text{ is null field and } F_1 \text{ is any field.} \tag{48}$$

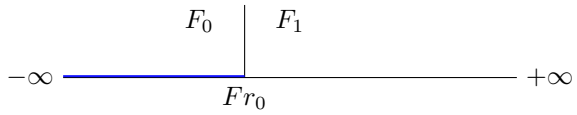

Figure 13: Truncated field $\mathrm{F}_{\mathrm{Truncated}}$ constitued of null field F_0, field F_1 and of frontier Fr_0.

• from a real Fr_1 allowing to generate the field F_1 on interval $]-\infty, Fr_1]$ and to generate the field F_2 on interval $[Fr_1, +\infty[$. Formally, field $\mathrm{F}_{\mathrm{Truncated}}$ presented in Figure 14 is a truncated field on $\mathbb{R}$ if and only if

$$\exists Fr_1 \in \mathbb{R}, \text{ such that: } \mathrm{F}_{\mathrm{Truncated}} |_{]-\infty, Fr_1]} = F_1, \mathrm{F}_{\mathrm{Truncated}} |_{[Fr_1, +\infty[} = F_2, \\ \text{where } F_1 \text{ is any field and } F_2 \text{ is null field.} \tag{49}$$

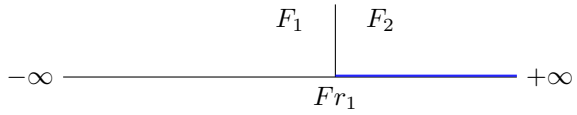

Figure 14: Truncated field $\mathrm{F}_{\mathrm{Truncated}}$ constitued of field F_1, null field F_2 and of frontier Fr_1.

The evaluation of truncated field $\mathrm{F}_{\mathrm{Truncated}}$ defined in relation (47) at instant t is given by following algorithm:

Algorithm 12: Evaluation algorithm of truncated field $\mathrm{F}_{\mathrm{Truncated}}$ defined in relation (47)

input : Fields F_0, F_1 and F_2, frontiers Fr_0 and Fr_1, inferior bound a and superior bound b
output: Evaluation value v
if $Fr_0 \leq t < Fr_1$ **then**
  v is the evaluation of field F_1 at instant t ;
else
  v is zero ;
return v ;

The evaluation of truncated field $\mathrm{F}_{\mathrm{Truncated}}$ defined in relation (48) at instant t is given by following algorithm:

Algorithm 13: Evaluation algorithm of truncated field $\mathrm{F}_{\mathrm{Truncated}}$ defined in relation (48)

input : Fields F_0 and F_1, frontier Fr_0, inferior bound a and superior bound b
output: Integration value v
if $t \geq Fr_0$ **then**
  v is the evaluation of field F_1 at instant t ;
else
  v is zero ;
return v ;

The algorithm for evaluation of truncated field $\mathrm{F}_{\mathrm{Truncated}}$ defined in relation (49) at instant t is given as follows:

Algorithm 14: Evaluation algorithm of truncated field $\mathrm{F}_{\mathrm{Truncated}}$ defined in relation (49)

input : Fields F_1 and F_2, frontier Fr_1, inferior bound a and superior bound b
output: Evaluation value v
if $t < Fr_1$ **then**
  v is the evaluation of field F_1 at instant t ;
else
  v is zero ;
return v ;

5 Operations between fields and measures

The discretization of field F_d in low level between inferior value x_a and superior value x_b is a task containing $\mathcal{N}_a^b + 1$ discrete values of a discrete field $(F_d^{\mathrm{D}}(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b}$ given by:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = F_d(y_k), \tag{50}$$

where points $(y_k)_{1 \leq k \leq \mathcal{N}_a^b + 1}$ are defined as:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], y_k = x_a + (k - 1) \times T_{\mathrm{dF}}. \tag{51}$$

5.1 Product of measure by a field

The interest payment measure is defined as the product of the loan rate measure by the current debt field. This induces that it is necessary to compute the product of measure by a field.

The product of discrete measure $(m_1(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b}$ by discrete field $(F_d^{\mathrm{D}}(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b}$ defined on the same universel mesh, is discrete measure $(m(n_a + j - 1))_{1 \leq j \leq \mathcal{N}_a^b}$ given by $\mathcal{N}_a^b$ values:

$$\forall k \in [\![1; \mathcal{N}_a^b]\!], m(n_a + k - 1) = m_1(n_a + k - 1) \times F_d^{\mathrm{D}}(n_a + k - 1). \tag{52}$$

The integration of discrete measure $(m(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b}$ defined in relation (52) between inferior bound a and superior bound b is the sum of values $(m(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b}$, which is given by following value:

$$\sum_{k=1}^{k=\mathcal{N}_a^b} m(n_a + k - 1). \tag{53}$$

5.2 Primitive of measure defined as a field

In the beginning we give some references to solve some ordinary differential equation in radon measure space. The key to integrate a real function or a measure is given in [9] as it's distribution function under a known domain. In particular in [19], we treat linear transport equation, in conservation form under weak regularity on the coefficients.

The Current Debt Field $\mathcal{K}_{RD}$ is related to Loan Measure $\tilde{\kappa}_E$ and Repayment Measure $\tilde{\rho}_{\mathcal{K}}$ by the following Ordinary Differential Equation:

$$\frac{d\mathcal{K}_{RD}}{dt} = \kappa_E(t) - \rho_{\mathcal{K}}(t). \tag{54}$$

where Loan Measure $\tilde{\kappa}_E = \kappa_E(t)dt$ is defined such that the amount borrowed between times t_1 and t_2 is:

$$\int_{t_1}^{t_2} \tilde{\kappa}_E, \tag{55}$$

and where repayment Measure $\tilde{\rho}_{\mathcal{K}} = \rho_{\mathcal{K}}(t)dt$ is defined such that the amount borrowed between times t_1 and t_2 is:

$$\int_{t_1}^{t_2} \tilde{\rho}_{\mathcal{K}}. \tag{56}$$

The solution of this ODE is expressed:

$$\mathcal{K}_{RD}(t) = \mathcal{K}_{RD}(t_{\mathrm{I}}) + \int_{t_{\mathrm{I}}}^{t} \tilde{\kappa}_E - \int_{t_{\mathrm{I}}}^{t} \tilde{\rho}_{\mathcal{K}}. \tag{57}$$

To compute the Current Debt Field $\mathcal{K}_{RD}$ at an instant t, we introduce the method which is computing the primitive of a measure. This method is based on numerical approach which consists in accumulating a discrete measure in order to approximate it by a piecewise function. Furthermore, since a primitive of measure m_d in low level, which is zero at point x_c is a field F_d, its discretization between inferior value x_a and superior value x_b with discrete step T_{dF} is defined by discrete field $(F_d^{\mathrm{D}}(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b + 1}$ given by:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = \int_{x_c}^{y_k} m_d, \tag{58}$$

where points $(y_k)_{1 \leq k \leq \mathcal{N}_a^b + 1}$ are defined in relation (50). We distinguish three cases of computing discrete field $(F_d^{\mathrm{D}}(n_a + k - 1))_{1 \leq k \leq \mathcal{N}_a^b + 1}$:

First case $x_c < x_a$

Measure m_d is discretized between points x_c and x_b with discrete step T_{dF} to compute discrete measure $(m_d(n_c + j - 1))_{1 \leq j \leq \mathcal{N}_c^b}$. The integral defined in relation (58) can be decomposed with Chasles relation to get:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = \sum_{j=1}^{j=\mathcal{N}_c^a} \int_{x_c + (j-1) \times T_{\mathrm{dF}}}^{x_c + j \times T_{\mathrm{dF}}} m_d + \sum_{j=1}^{j=k-1} \int_{x_a + (j-1) \times T_{\mathrm{dF}}}^{x_a + j \times T_{\mathrm{dF}}} m_d. \tag{59}$$

Replacing x_a by $x_c + \mathcal{N}_c^a \times T_{\mathrm{dF}}$ in relation (59), we obtain the following equality:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = \sum_{j=1}^{j=\mathcal{N}_c^a} \int_{x_c + (j-1) \times T_{\mathrm{dF}}}^{x_c + j \times T_{\mathrm{dF}}} m_d + \sum_{j=1}^{j=k-1} \int_{x_c + (j-1+\mathcal{N}_c^a) \times T_{\mathrm{dF}}}^{x_c + (j+\mathcal{N}_c^a) \times T_{\mathrm{dF}}} m_d. \tag{60}$$

From this and using relation (13) which defines discrete measure, we get:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = \sum_{j=1}^{j=\mathcal{N}_c^a} m_d(n_c + j - 1) + \sum_{j=1+\mathcal{N}_c^a}^{j=k-1+\mathcal{N}_c^a} m_d(n_c + j - 1). \tag{61}$$

Second case $x_c > x_b$

Measure m_d is discretized between points x_a and x_c with discrete step T_{dF} to compute discrete measure $(m_d(n_a + j - 1))_{1 \leq j \leq \mathcal{N}_a^c}$. The integral defined in relation (58) can be decomposed with Chasles relation

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = -\sum_{j=k}^{j=\mathcal{N}_a^c} \int_{x_a + (j-1) \times T_{\mathrm{dF}}}^{x_a + j \times T_{\mathrm{dF}}} m_d. \tag{62}$$

From this, we get:

$$\forall k \in [\![1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = -\sum_{j=k}^{j=\mathcal{N}_a^c} m_d(n_a + j - 1). \tag{63}$$

Third case $x_a \leq x_c \leq x_b$

We determine integer $L \in [\![1; \mathcal{N}_a^b]\!]$ satisfying following inequalities:

$$y_L < x_c \leq y_{L+1}. \tag{64}$$

Since $x_c > y_k$ for k from 1 to L, the result of the second case gives:

$$\forall k \in [\![1; L]\!], F_d^{\mathrm{D}}(n_a + k - 1) = -\sum_{j=k}^{j=\mathcal{N}_a^c} m_d(n_a + j - 1). \tag{65}$$

Replacing x_c by $x_a + \mathcal{N}_a^c \times T_{\mathrm{dF}}$, the integral defined in relation (58) can be decomposed with Chasles relation to get:

$$\forall k \in [\![L + 1; \mathcal{N}_a^b + 1]\!], F_d^{\mathrm{D}}(n_a + k - 1) = \sum_{j=1+\mathcal{N}_a^c}^{j=k-1} m_d(n_a + j - 1). \tag{66}$$

6 Architecture

This section gives concrete implementation details for the implemented API and examples of its usage. An excessive amount of work was done to have a decent precondition validation and this section mostly focuses on the API aspect of this implementatition

Naturally, after description of the different algorithm and computation in financial model, we develop API to attempt the objectives: more efficient computation and robustness.

6.1 Interfaces

The computation in library Lemf is defined as a graph where the terminal nodes are measures or fields, and where the non-terminal nodes are operations. Consequently, Lemf allows to build its dynamically. We realize these graphs by abstract factory class called "AbstractFactor".

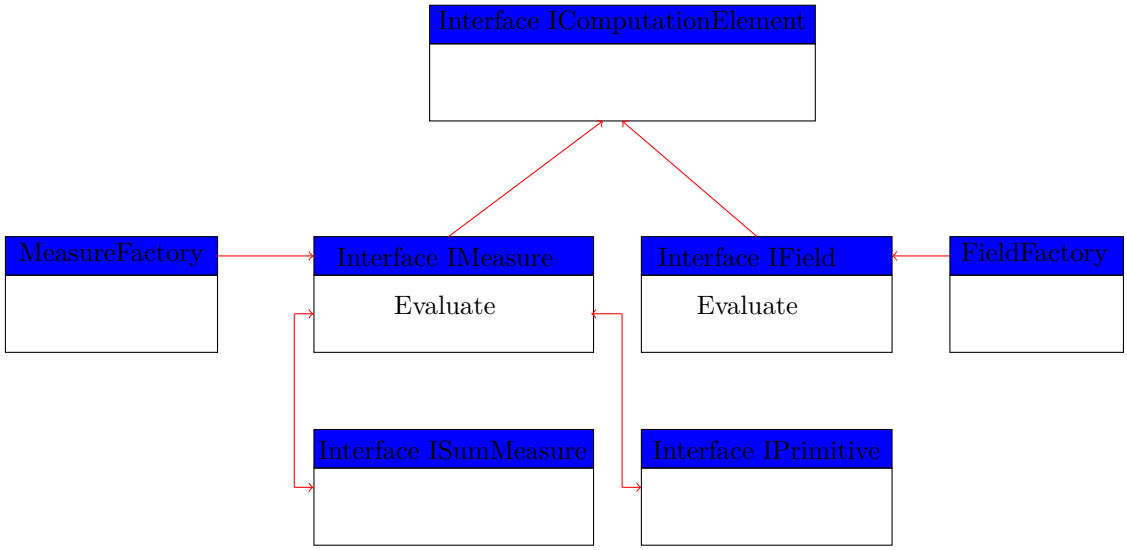

Figure 15: Class diagrams of library Lemf.

In AbstractFactor an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern. This factory AbstractFactor creates measures, fields and operations which are used by interfaces.

We describe some interfaces used in library LemfAN. The first interface IMeasureAN implements all measures in low level and provides a method EvaluateDiscrete allowing to discretize a non-discrete measure between inferior bound a and superior bound b with minimal observation step $T_{\min}$. This method EvaluateDiscrete returns a task containing $\mathcal{N}_a^b$ discrete values.

```
interface IMeasureAN
{
Task<IEnumerable<double>> EvaluateDiscrete(decimal a, decimal b, decimal T_min) {}
}
```

The second interface IFieldAN is defined by the evaluation interface of field. IFieldAN contains a method EvaluateDiscrete which is allowing to evaluate a field between inferior value a and superior value b with discrete step T_{dF}. This method EvaluateDiscrete returns a task containing $\mathcal{N}_a^b + 1$ discrete values.

```
interface IFieldAN
{
Task<IEnumerable<double>> EvaluateDiscrete(decimal a, decimal b, decimal T_dF) {}
}
```

There are some principal interfaces used in library Lemf. The first interface IMeasure implements all measures in high level. It provides a method Evaluate allowing to integrate a measure between inferior bound a and superior bound b in order to return a value.

```
interface IMeasure
{
double Evaluate(decimal a, decimal b) {}
}
```

The second interface IField is defined by the evaluation interface of field. We notice that interface

IField implements all fields in high level. This interface IField contains a method Evaluate which is allowing to evaluate a field at an instant v returning a value.

```
interface IField
{
double Evaluate(decimal v) {}
}
```

We create interface IComputationElementAN to encompass the notion of field and measure in only one interface defined as:

```
public interface IComputationElementAN
{
IEnumerable<IComputationElementAN> Children
{
get;
}
}
```

We developped for each measure and for each field their own interfaces. For instance, interface IAffineMeasure defined by affine measure interface inherits from IMeasureAN. This interface IAffineMeasure uses attributes C_1 and C defining affine measure in table 1.

```
public interface IAffineMeasure : IMeasureAN
{
double C
{
get;
}

double C1
{
get;
}
}
```

6.2 Abstract classes

An abstract class is a class whose the implementation is not complete and that is not instanciable. It is the base for other derived classes (inherited). We created abstract classes in the library LemfAN as AbstractLeafMeasureAN to characterize simple measures, AbstractComposedMeasureAN to characterize composed measures and AbstractMeasureAN to characterize simple and composed measures. Abstract class AbstractLeafMeasureAN inherits from abstract class AbstractMeasureAN which means that each simple measure is a measure. Abstract class AbstractComposedMeasureAN inherits from abstract class AbstractMeasureAN which means that each composed measure is a measure.

We notice that AbstractMeasureAN characterizes a general concept integration of measure in low level defined in section 2. Indeed, it contains a method EvaluateDiscrete which sum a discrete measure on each observation step T_{obs}.

Since we justified the creation of abstract classes AbstractLeafMeasureAN, AbstractComposedMeasureAN and AbstractMeasureAN, we created for the same reasons the following abstract classes in Lemf which are AbstractLeafMeasure, AbstractComposedMeasure and AbstractMeasure.

6.3 Unit tests

Unit test [4] is a software testing method by which individual units of our libraries LemfAN and Lemf. For each measure and field, we constitue sets of one or more computer program modules in order to determine whether they are fit for use.

Paper [8] shows why that Team System is written for any software team that is considering running a software project using code coverage under Visual Studio. Unit test cover a significant proportion of our librairies. For this, we used the test coverage (see Figure 18) to determine the proportion of our code library that will be really tested by coded tests. It allows to provide effective protection against bugs.

6.4 Math.NET Numerics

Math.NET Numerics [1] aims to provide methods and algorithms for numerical computations in science, engineering and every day use. Covered topics include special functions, linear algebra, probability models, random numbers, interpolation, integration, regression, optimization problems and more.

Being written in it, Math.NET Numerics works very well with *C#* and related .Net languages. For implementing the convolution operator in *C#*, we used the Math.NET Numerics [1] IntegralTransforms library. In particular, we use expression Invoke to express actions that must run simultaneously.

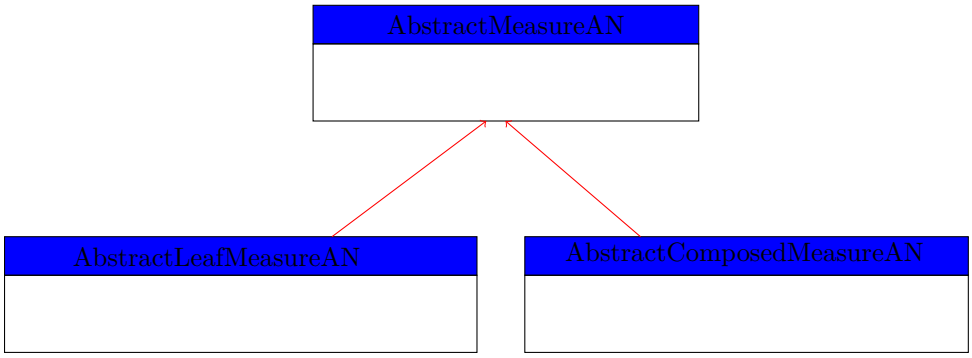

Figure 16: Abstract Class diagrams of library LemfAN.

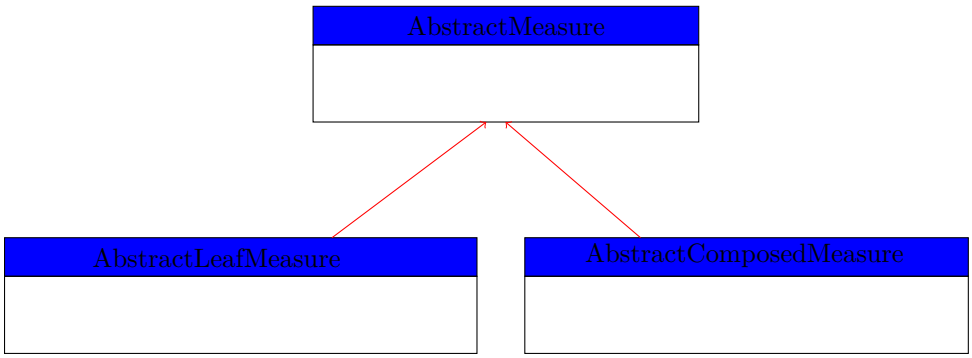

Figure 17: Abstract Class diagrams of library Lemf.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
lhelard-d_AFI-PC 2013-12-09 15_14_48.coverage	170	3,60 %	4553	96,40 %
expressionparser.dll	28	22,40 %	97	77,60 %
mathembedded.dll	65	12,10 %	472	87,90 %
MathEmbedded	65	12,10 %	472	87,90 %
AbstractLeafMeasure	9	100,00 %	0	0,00 %
get_Children()	9	100,00 %	0	0,00 %
AffineMeasure	0	0,00 %	20	100,00 %
ApproximateDoubleComparer	3	42,86 %	4	57,14 %
Compare(object, object)	3	42,86 %	4	57,14 %
ConstantMeasure	0	0,00 %	14	100,00 %
CosinusMeasure	0	0,00 %	29	100,00 %
DiracMeasure	0	0,00 %	23	100,00 %
ErrorApproxiamtion	0	0,00 %	3	100,00 %
Extensions	0	0,00 %	16	100,00 %
GaussianMeasure	0	0,00 %	18	100,00 %
GaussianMeasure.<>c__DisplayClass2	0	0,00 %	5	100,00 %
IdentityMeasure	14	100,00 %	0	0,00 %
Evaluate(decimal, decimal)	9	100,00 %	0	0,00 %
IdentityMeasure(MathEmbedded.IMeasure)	2	100,00 %	0	0,00 %
get_Measure()	2	100,00 %	0	0,00 %
set_Measure(MathEmbedded.IMeasure)	1	100,00 %	0	0,00 %
IdentityMeasure.<get_Children>d__0	2	100,00 %	0	0,00 %
NullMeasure	0	0,00 %	10	100,00 %
PiecewiseMeasure	0	0,00 %	53	100,00 %
PolynomialMeasure	0	0,00 %	22	100,00 %
QuadraticMeasure	0	0,00 %	24	100,00 %
RescaledMeasure	0	0,00 %	15	100,00 %
RescaledMeasure.<get_Children>d__0	3	100,00 %	0	0,00 %
MoveNext()	3	100,00 %	0	0,00 %
ShiftMeasure	3	10,34 %	26	89,66 %
ShiftMeasure.<get_Children>d__0	3	100,00 %	0	0,00 %
MoveNext()	3	100,00 %	0	0,00 %
SinusMeasure	0	0,00 %	29	100,00 %
StepMeasure	0	0,00 %	20	100,00 %
SubstractionMeasure	0	0,00 %	12	100,00 %
SubstractionMeasure.<get_Children>d__0	3	100,00 %	0	0,00 %
MoveNext()	3	100,00 %	0	0,00 %
SumMeasure	13	18,06 %	59	81,94 %
Add(MathEmbedded.IMeasure)	3	17,65 %	14	82,35 %
Evaluate(decimal, decimal)	0	0,00 %	15	100,00 %
Remove(MathEmbedded.IMeasure)	10	30,30 %	23	69,70 %
SumMeasure(MathEmbedded.IMeasure[])	0	0,00 %	3	100,00 %
SumMeasure(System.Collections.Generic.ICollectio...	0	0,00 %	2	100,00 %
get_Children()	0	0,00 %	2	100,00 %
TabulatedMeasure	9	20,93 %	34	79,07 %
Evaluate(decimal, decimal)	9	100,00 %	0	0,00 %
Tabulate(decimal, decimal, System.Collections.Gen...	0	0,00 %	32	100,00 %
TabulatedMeasure()	0	0,00 %	2	100,00 %
TruncatedMeasure	0	0,00 %	36	100,00 %
TruncatedMeasure.<get_Children>d__0	3	100,00 %	0	0,00 %
MoveNext()	3	100,00 %	0	0,00 %

Figure 18: Coverage at $96,4\%$ of test library LemfTest.

A Simple measures

Simple measures	Definition	Value of integration
Null measure	$\tilde{m}_{\text{Null}} = 0$	0
Affine measure	$\forall t \in \mathbb{R}, \tilde{m}_{\text{Affine}} = (\text{C}_1 \times t + \text{C}) \times \lambda_{\text{Lebesgue}}$	$\frac{\text{C}_1}{2} \times (b^2 - a^2) + \text{C} \times (b - a)$
Quadratic measure	$\forall t \in \mathbb{R}, \tilde{m}_{\text{Quadratic}} = (\text{C}_2 \times t^2 + \text{C}_1 \times t + \text{C}) \times \lambda_{\text{Lebesgue}}$	$\frac{\text{C}_2}{3} \times (b^3 - a^3) + \frac{\text{C}_1}{2} \times (b^2 - a^2) + \text{C} \times (b - a)$
Polynomial measure	$\forall t \in \mathbb{R}, \tilde{m}_{\text{Polynom}} = \left(\sum_{i=0}^{i=n} \text{C}_i \times t^i \right) \times \lambda_{\text{Lebesgue}}$	$\sum_{i=0}^{i=n} \frac{\text{C}_i}{i+1} \times \left(b^{i+1} - a^{i+1} \right)$
Sinus measure	$\forall t \in \mathbb{R}, \tilde{m}_{\text{Sinus}} = \sin(\text{C}_1 \times t + \text{C}) \times \lambda_{\text{Lebesgue}}$	$(b - a) \times \sin(\text{C})$, if $\text{C}_1 = 0$ $\frac{\cos(\text{C}_1 \times a + \text{C}) - \cos(\text{C}_1 \times b + \text{C})}{\text{C}_1}$, if $\text{C}_1 \neq 0$
Cosinus measure	$\forall t \in \mathbb{R}, \tilde{m}_{\text{Cosinus}} = \cos(\text{C}_1 \times t + \text{C}) \times \lambda_{\text{Lebesgue}}$	$(b - a) \times \cos(\text{C})$, if $\text{C}_1 = 0$ $\frac{\sin(\text{C}_1 \times b + \text{C}) - \sin(\text{C}_1 \times a + \text{C})}{\text{C}_1}$, if $\text{C}_1 \neq 0$
Exponential measure	$\forall t \in \mathbb{R}, \tilde{m}_{\text{Exponential}} = \text{e}^{\text{C} \times t} \times \lambda_{\text{Lebesgue}}$	b-a, if $\text{C} = 0$ $\frac{\text{e}^{\text{C} \times b} - \text{e}^{\text{C} \times a}}{\text{C}}$, if $\text{C} \neq 0$
Dirac measure	$\tilde{m}_{\text{Dirac}}$ in point L, and mass M	M, if $a \leq L < b$ 0, if $L < a$ or $b \leq L$

Table 1: The integration of some simple measures

B Simple fields

Simple fields	Definition	Value of evaluation at instant d
Null field	$\forall t \in \mathbb{R}, F_{\text{Null}}(t) = 0$	0
Affine field	$\forall t \in \mathbb{R}, F_{\text{Affine}}(t) = \text{C}_1 \times t + \text{C}$	$\text{C}_1 \times d + \text{C}$
Quadratic field	$\forall t \in \mathbb{R}, F_{\text{Quadratic}}(t) = \text{C}_2 \times t^2 + \text{C}_1 \times t + \text{C}$	$\text{C}_2 \times d^2 + \text{C}_1 \times d + \text{C}$
Polynomial field	$\forall t \in \mathbb{R}, F_{\text{Polynom}}(t) = \sum_{i=0}^{i=n} \text{C}_i \times t^i$	$\sum_{i=0}^{i=n} \text{C}_i \times d^i$
Sinus field	$\forall t \in \mathbb{R}, F_{\text{Sinus}}(t) = \sin(\text{C}_1 t + \text{C})$	$\sin(\text{C}_1 d + \text{C})$
Cosinus field	$\forall t \in \mathbb{R}, F_{\text{Cosinus}}(t) = \cos(\text{C}_1 t + \text{C})$	$\cos(\text{C}_1 d + \text{C})$
Exponential field	$\forall t \in \mathbb{R}, F_{\text{Exponential}}(t) = \text{e}^{\text{C} \times t}$	$\text{e}^{\text{C} \times d}$

Table 2: The evaluation of some simple fields

Conflict of Interests

The authors declare that there is no conflict of interests.

References

[1] Math.net numerics[online]. numerics.mathdotnet.com. Accessed: 2016-06-30.

[2] Sofi [online]. https://www.mgdis.fr/index.php?page=display_doma&class=article&object=sol_sofi_programmation_financiere&method=display_full&refo=001009. Accessed: 2016-08-16.

[3] Task parallel library [online]. https://msdn.microsoft.com/fr-fr/library/dd537609%28v=vs.110%29.aspx. Accessed: 2016-06-30.

[4] Unit test [online]. https://msdn.microsoft.com/en-us/library/hh694602.aspx. Accessed: 2016-08-16.

[5] Sterling-K Berberian. Measure and integration. 1965.

[6] Tarik Chakkour and Emmanuel Frénod. Inverse problem and concentration method of a continuous-in-time financial model. *International Journal of Financial Engineering*, 3(No 2):20, 2016.

[7] Jintai Chung and Jang Moo Lee. A new family of explicit time integration methods for linear and non-linear structural dynamics. *International Journal for Numerical Methods in Engineering*, 37(23):3961–3976, 1994.

[8] Robert DeLine and Kael Rowan. Code canvas: zooming towards better development environments. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 207–210. ACM, 2010.

[9] Dieter Denneberg. *Non-additive measure and integral*, volume 27. Springer Science & Business Media, 1994.

[10] Emmanuel Frénod and Tarik Chakkour. A continuous-in-time financial model. *Mathematical Finance Letters*, pages 1–36, ID2, 2016.

[11] Emmanuel Frénod, Pierre Menard, and Mohamad Safa. Optimal control of a continuous-in-time financial model. *Mathematical Modelling and Numerical Analysis*, 2013. Manuscript in revision.

[12] Emmanuel Frénod, Pierre Menard, and Mohamad Safa. Two optimization problems using a continuous-in-time financial model. *Journal of Industrial and Management Optimization*, 2014. Manuscript in revision.

[13] Emmanuel Frénod and Mohamad Safa. Continuous-in-time financial model for public communities. volume 45, pages 158–167. EDP Sciences, 2014.

[14] Gusein Sh Guseinov. Integration on time scales. *Journal of Mathematical Analysis and Applications*, 285(1):107–127, 2003.

[15] Thomas Hahn. CubaâĂŤa library for multidimensional numerical integration. *Computer Physics Communications*, 168(2):78–95, 2005.

[16] Tristan A Hearn and Lothar Reichel. Fast computation of convolution operations via low-rank approximation. *Applied Numerical Mathematics*, 75:136–153, 2014.

[17] Daan Leijen, Wolfram Schulte, and Sebastian Burckhardt. The design of a task parallel library. *Acm Sigplan Notices*, 44(10):227–242, 2009.

[18] Rainer Machné, Andrew Finney, Stefan Müller, James Lu, Stefanie Widder, and Christoph Flamm. The sbml ode solver library: a native api for symbolic and fast numerical analysis of reaction networks. *Bioinformatics*, 22(11):1406–1407, 2006.

[19] F Poupaud and M Rascle. Measure solutions to the linear multi-dimensional transport equation with non-smooth coefficients. *Communications in Partial Differential Equations*, 22(1-2):225–267, 1997.

[20] Toshiya Tanaka, I Tomohiro, Shunsuke Inenaga, Hideo Bannai, and Masanori Takeda. Computing convolution on grammar-compressed text. In *Data Compression Conference (DCC), 2013*, pages 451–460. IEEE, 2013.

[21] Alan J Weir. *General integration and measure*, volume 2. CUP Archive, 1974.